

# **LISTing**

## **Newsletter**

Newsletter of the  
Long Island Sinclair/Times  
Users Group  
.....

Incorporating NYTSE

January, 1988

## **"SPECIAL TECHIE ISSUE"**

**IN this issue:** hardware projects, and MC programming...why?  
Find out by reading 'Newsletter Newsnotes'.

## **NEXT MEETINGS: IMPORTANT!**

The GUY FROM TIMEX! OFFICER ELECTIONS! DUES DUE!! All at Harvey's  
house, Feb. 21. Call 516-791-6247 for more info.  
NYTSE meets Mon. Feb. 22, 7PM, Miss Kim's, Park Ave. S.  
PLEASE NOTE: Meetings are 3rd Sunday + Mon. between 21st<sup>in</sup> Feb. and 22nd St.

**L.I.S.T.**  
**5 Peri Lane**  
**Valley Stream, NY 11581**

**TO:**

**FIRST CLASS MAIL**  
**DATE MEETING NOTICE**  
**Please DON'T delay!**

=====

LISTing Listing

Please send submissions to:

Joe Newman, 325 W. Jersey St.,

#2D, Elizabeth, NJ 07202

or send items for the LIST group

to: LIST, Harvey Rait

5 Peri Lane

Valley Stream, NY 11581

PLEASE NOTE THE NEW LIST ADDRESS

yearly LIST dues- \$15

=====

#### URGENT BUSINESS!!!

ONE- it's time to elect new officers of LIST. Show up at the February meeting if you wish to participate in the election process- hopefully to become an officer!

TWO- most important- DUES ARE DUE! Send in your yearly dues NOW! Keep the TS spirit alive! Support your systems- support LIST. Make \$15 checks or money orders payable to LIST.

#### NEWSLETTER NEWSNOTES

Why a 'technical issue'? Well the main reason is that I'm stuck with the Editor's Dilemma again- yep, no more articles. Oh I have a few, but they're more hardware projects. So bust out the Tasword, fire up the Quill, or if you must- ink up the pen and SEND IN SOME MORE ARTICLES! I truly thank all of you who have submitted and continue to submit articles. On the most part the articles have been of top quality- keep up the good work!

As for the meetings- as the front page says "The man from TIMEX will be at the Feb. meeting"!!! Actually he is Mr. Skyrme, President of the American office of Psion, Inc. His purpose is to show off the Psion Organizer devices, but we know better. We'll let him show us his gadget, then we'll zap him with those TS questions! Mr. Skyrme was the Product Development Director for the 2068 at Timex.

Mr. Skyrme is also scheduled to speak at the upcoming Trenton Computer Fair. He will be speaking at our T/S

representation, scheduled for April 23 in the afternoon. As in previous years, our groups will be present at the fair to show Sinclair is still going good. This fair draws T/S people from up & down the east coast, so make plans to attend...April 23 & 24.

Nyles is doing well, and may make it to the March meeting. If any of you tried to contact him at the address printed in the last issue, my apologies if you had any trouble reaching him. Nyles is no longer at that particular spot. Last I know he was in the V.A. Hospital, 1st Ave. and 24th Street, ward 8 North, Bed 24.

If anyone is interested in using a bar code reader with the QL, please contact me. I have a working point-of-sale database system for the QL, which is used with a bar code wand. I also have completed a program which can print bar code labels on an Epson compatible printer (i.e., the QL Printer, which uses double density bit graphics). The program can print bar codes of all upper case alpha characters and the numbers. By the way, the p-o-s program is a full featured program comparable with those on 'other' machines. Also, I have a cash register inventory program for the 2068, and plan to release one for the QL which does not require the bar code wand, and also a generic bar code database which can be customized by the end user for a variety of applications. Again, contact me if you're interested.

Bill Jones of the TS-2068 Update says he needs a Telecomputing writer and someone to keep going with extra memory programming for the 2068. Contact him if you'd like to get involved at:

TS-2068 Update  
1317 Stratford Avenue  
Panama City, FL 32404

A common complaint of many TIMEX equipment owners is the lack of a power on LED (light emitting diode) on the equipment. The following paragraphs describe how to add a LED to a TIMEX 2040 or ALPHACOM 32 thermal printer. It is a simple project but be advised that the author and the LIST GROUP are not responsible for any damage you may do to your equipment.

The parts required for this project are a LED of your choice (almost any color or size will do as long as it will fit into the printer's case), one 470 ohm 1/8 to 1/2 watt resistor, a short length of "telephone" or wire wrap wire and a bit of rosin core solder. The tools required are a Phillips head screwdriver, wire cutters or knife, a small soldering iron and drill for the LED mounting hole.

The first step is to open the printer's case. To do this remove the paper from the printer, turn the case over and remove the four screws holding the two halves together. Turn the printer upright and lift the top off. The PC board and printer mechanism should remain attached to the bottom of the case. Next drill a hole to mount the LED. The front right hand corner of the case seems appropriate but it can be mounted anywhere there is room inside for the LED's leads. Be sure the case will close BEFORE you drill, as the saying goes "Measure twice, cut once."

Once the case is prepared, the LED and resistor can be connected. Refer to FIGURE ONE for the connection locations. Solder one end of the 470 ohm resistor to the jumper wire marked "W2", then a length of wire long enough to reach the mounted LED should be soldered to the other end of the resistor. Solder the end of this wire to the anode of the LED. The anode lead on round LED's has a small flat spot next to it on the "mounting rim". If you are unsure of the proper connection, solder it to either lead. Now solder a length of wire from the remaining lead to the right side of diode 5 on the PC board (marked D5). Connect the power to the printer with the cover off to test the LED. If the diode fails to light with the connections complete, unsolder the two wires to the diode's leads, switch them around and solder again. Test the LED again by connecting the power. If it still fails to light, check the solder joints for a proper connection or replace the diode in case it is defective.

The last step is to reassemble the case. Close the cover making sure none of the wires will interfere with the printer mechanism, the switches or mounting screws. Reinstall the four screws taking care not to strip the plastic threads in the case. Insert the paper and the printer is ready for use.

VIEW OF THE TOP  
OR COMPONENT  
SIDE OF PRINTER  
P.C. BOARD

CONNECT RESISTOR  
AND LED BETWEEN  
W2 AND D5. NOTE  
POLARITY OF LED  
AS DESCRIBED IN  
ARTICLE.

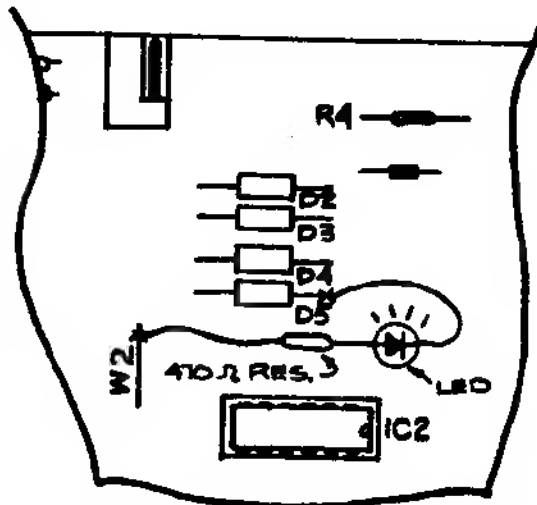


FIGURE ONE

I CAN'T BELIEVE  
IT!... THIS  
GUY STILL  
USES A  
TIMEX!

## MAKING CODE RELOCATABLE

by J. Leary

Most machine code programs for T/S computers will only run from a single location in RAM. This is because programs usually have absolute addresses imbedded in them, and is a source of frustration. If 2 programs overlap, for example, they can't both reside in memory, and so can't both be used.

Here is one way of dealing with the problem.

The program MR1BT (Make Relative-1 Byte) will modify another machine code program so that it will run at any location in RAM. It is only 60 bytes long but requires you to construct a data table.

The approach is that in a program with many jumps, calls and loads to/from absolute addresses, most such locations will be less than 256 bytes apart. A table of 1-byte offsets can be used to store these separations.

Because offsets are at least 2 bytes apart, 00 and 01 are free to be used as flags: 00 signals end of table and 01 marks a 2-byte offset. (see next paragraph)

There is a penalty to be paid if successive locations are more than 255 bytes apart. The table must then contain 3 bytes for this next location:

01 to warn 2-byte offset coming  
2-byte offset (LO-byte first).

Your program must be entered by a USR call from Basic, so that the address called is in the BC register. I believe all Timex/Sinclair computers are alike in this respect.

Proceed as follows:

1. Prefix your program with the table constructed as follows:

1st 2 bytes-starting address of your program including my MR1BT.  
next bytes-offsets as described above; 1 byte for locations < 256 bytes apart; 3 bytes for locations more than 256 bytes apart.  
last byte-00 to mark end of table.

2. Prefix the table with the MR1BT program below. If your program does not run from its first byte, insert a jump to the running address at the end of MR1BT. This will become the first offset in your table.

3. To relocate the program, load to any address a; then just RANDOMIZE USR a. (or PRINT USR a, whichever your program uses). All absolute addresses marked by the table are now adjusted to the new location. Also the address a has been put into the table (1st 2 bytes). And your program is up and running.

4. If you call various parts of the program from Basic, the calls will need adjusting. If you use a variable for the calls (with the value of a above), then you need make only one adjustment.

5. It is not easy to get all the offsets right. One error or omission and everything comes out wrong for the rest of your program. So MR1BT has no commercial prospects, and I offer it for free to the T/S Community.

#### MR1BT

ADDR	OP	CODE	Mnemonics	Commentary
C000	C5		PUSH BC	SAVE USR CALL
C001	213D00		LD HL,003D	
C004	09		ADD HL,BC	ADD OFFSET TO USR CALL
C005	4E		LD C,(HL)	
C006	23		INC HL	1ST 2 BYTES OF TABLE INTO BC;
C007	46		LD B,(HL)	THIS IS THE OLD USR CALL
C008	2B		DEC HL	
C009	E3		EX (SP),HL	
C00A	E5		PUSH HL	
C00B	AF		XOR A	CARRY FLAG=0
				x=DIFF. OF 2 USR CALLS/ALL ABS.
				ADDRESSES MUST BE ADJ. BY X
C00C	ED42		SBC HL,BC	
C00E	C1		POP BC	
C00F	E3		EX (SP),HL	
C010	71		LD (HL),C	PUT NEW USR CALL IN 1ST 2 BYTES
C011	23		INC HL	OF TABLE/THIS FEATURE MAKES
C012	70		LD (HL),D	PROPER RELOCATION AUTOMATIC
C013	23		INC HL	LOOP STARTS HERE
C014	7E		LD A,(HL)	A=NEXT TABLE ENTRY
C015	5F		LD E,A	
C016	FE00		CF 00	IF 0, END OF TABLE/JUMP TO YOUR
C018	281F		JR Z,C039	PROGRAM
C01A	FE01		CP 01	1 BYTE PENALTY
C01C	2804		JR Z,C022	APPLIES
C01E	1600		LD D,00	DOED NOT APPLY
C020	1804		JR C026	
C022	23		INC HL	
C023	5E		LD E,(HL)	PUT OFFSET INTO DE

C024	23	INC HL	
C025	56	LD D,(HL)	
MAIN ADJUSTING ROUTINE			
C026	E3	EX (SP),HL	PUT X IN HL & TABLE PLACE ON STACK
C027	EB	EX DE,HL	X IN DE;TABLE OFFSET IN HL
C028	09	ADD HL,BC	PROGRAM LOCATION (PL) IN HL
C029	4E	LD C,(HL)	CONTENTS OF PL, NEXT ADDRESS YO
C02A	23	INC HL	BE ADJUSTED (NA) IN BC
C02B	46	LD B,(HL)	I.E. NA=(PL)
C02C	C5	PUSH BC	X ON STACK
C02D	E3	EX (SP),HL	NA IN HL;PL+1 ON STACK
C02E	19	ADD HL,DE	ADJUST NA
C02F	EB	EX DE,HL	
C030	E3	EX (SP),HL	X ON STACK;PL+1 IN HL
C031	72	LD (HL),D	
C032	2B	DEC HL	PUT ADJUSTED ADDRESS
C033	73	LD (HL),E	INTO PROGRAM
C034	E3	EX (SP),HL	X IN HL;PL ON STACK
C035	C1	POP BC	PL IN BC
C036	E3	EX (SP),HL	X ON STACK;TABLE PLACE IN HL
C037	18DA	JR C013	LOOP
C039	D1	POP DE	EXIT TO RELOCATED PROGRAM;
C03A	E9	JP (HL)	(IF NEEDED ENTER AN
C03B	00	NOP	ABSOLUTE JUMP IN THESE
C03C	00	NOP	LAST 3 BYTES)

Here is an example of how it works. Consider this little program which prints a text (not shown).

F000		(256 BYTES OF TEXT)
F100	FF	END OF TEXT
F101	2100F0	LD HL,F000
F104	7E	LD A,(HL)
F105	FEFF	CP FF
F107	2B04	JR Z,F10D
F109	D7	RST 10h
F10A	23	INC HL
F10B	1BF7	JR F104
F10D	C9	RET.

To make this program relocatable, note there is only one absolute address, and the text must be jumped over. The table will be 7 bytes long, 2 for the new start, 1 byte offset to the jump, 3 byte offset to the address at F102, and 00 to end the table.

Add the 7 byte table to the 60 byte MR1BT program and you get 43h. So MR1BT is located at F000-0043=EFBD. At EFF6, a jump to F101 must be inserted (else JP (HL) puts you in a crash at F000, the text). The first offset is EFF7-EFBD=003A, and the next is F102-EFF7=010B.

Here is how your relocatable program looks:

ADDR OF CODE	MNEMONICS	COMMENTARY
--------------	-----------	------------

EFBD C5	PUSH BC	START OF MR1BT
****		
EFF6 C301F1	JP F101	JUMP OVER TEXT
	START OF TABLE	
EFF9 BDEF		start address,lo byte first
EFFB 3A		1st offset
EFFC 010B01		2-byte offset,lo byte first,with 01 flag
FFFF 00		end of table
F000-F10D		PROGRAM AS ABOVE

\*\*\*\*\*  
 \* USER DEFINED GRAPHICS, REVISITED! \*  
 \*\*\*\*\*  
 BY CEDRIC R. BASTIAANS

My article "USER DEFINED GRAPHICS MADE EASY" (LISTing of September '87 and HATS of October '87) requires a few further observations...

(1) When the program has been loaded and it runs, one of the first messages on screen is that the UDG's cannot get lost by such commands as NEW, CLEAR, SAVE or LOAD. Even though this is true, it might suggest that the UDG's can be SAVED! The only way to SAVE UDG's is by the method shown in the second page of my article, by DATA, READ and RESTORE commands.

(2) Then, Basil Wentworth brought to my attention that it is possible for the program to run into a snag. That happens when the BLACK SQUARES occuring in program lines 120, 210 and 230 are constructed by:

INVERSE or INVERSE VIDEO "space"

Should you have done this, you will have noticed that you can't progress beyond line 120, since you will get the error message:

K Invalid color, 120:3

I would never have thought of using any of the above commands to make a black square, so I never ran into this problem. Two correct ways of making a black square are:

- (a) GRAPHICS shifted 8 (then undo the graphics mode)
- or
- (b) CHR\$143

Method (a) is by far the easiest.

I hope that you have not been discouraged by any such problem; try the program and you will find that it is the easiest way yet to make UDG's!

## ZY81/TSXXXX OUT-BOARD REAL-TIME CLOCK

by Tim Stoddard

The following is a description of the Real-Time clock section of the D.A.M. board article which appeared in Time Designs Magazine ( March/April '87 page 23 ). The board features a full-function real-time clock with battery back-up and is mappable into eight I/O port blocks. See attached schematic diagram.

### 8255 PPI

The 8255 PPI is used to interface the MSM5832 clock to the Z80 buss. This is needed because the MSM5832 is too slow of a device to attach directly to the Z80 buss.

The 8255 has three ports called "A", "B", and "C". The module uses port "A" to xfer data to the MSM5832, port "B" to select the proper register in the MSM5832, and port "C" to toggle the control signals to the MSM5832. These three ports are accessed by writing or reading one of four registers in the 8255. Port "C" is also bit-addressable by writing a special code via the CONTROL register. This allows us to toggle a single control signal to the MSM5832 without having to "remember" what state the other three MSM5832 control signals were. The following is the table showing which registers control what in the 8255:

REGISTER	PORT
0	A
1	B
2	C
3	CONTROL

The first action that should occur in your BASIC or machine language routine is to initialize the 8255 by clearing port "C" and then setting up the 8255 to write, or read the MSM5832 clock chip. This is done via register 3; the CONTROL register. To write to the MSM5832 you need to initialize the 8255 so that ports A, B, and C are all "output" ports. This is done by writing the control register in the 8255 with an 80<sub>HEX</sub> ( 128<sub>DEC</sub> ). To read the MSM5832 we'll need to initialize the 8255 so that port A is an input port and ports B, and C are output ports. This is done by writing the CONTROL register in the 8255 with a 90<sub>HEX</sub> ( 144<sub>DEC</sub> ). A one-time action after powering up the computer is to initialize the 8255 ( to read or write the MSM5832 ) then "clear" port "C" by writing a ZERO to register 2. This resets the active-high control signals HOLD, RD, and WR to the MSM5832, and causes the MSM5832's CS to go active via the inverter.

The 74HC138 is used to select a "block" of I/O ports to be used with the clock module. The HEX address of the ports always end in 3, 7, B, or F. So if the 74HC138 is strapped



for "block five", the I/O ports would be 53, 57, 5B, and 5F for 8255 registers 0, 1, 2, and 3, respectively. The following table summarizes the relationships:

8255 PORT	8255 reg ADDRESS	TIMEY		MSM5832 FUNCTION
		HEX	DEC	
A	0	53	83	DATA port
B	1	57	87	REGISTER ADDRESS port
C	2	5B	91	CONTROL port
CNT	3	5F	95	-----

The above table assumed a "block" of 5. You can re-strap to any of the eight blocks to avoid conflict with other devices attached to your computer.

We use a special function in the 8255 that allows us to change single bits in port "C" rather than "remembering" what all the bits should be in port "C". This is done by writing a special code via the 8255's CONTROL port 5F, i.e. The syntax of this code is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	X	X	X	SR

SELECTED  
BIT

000 = 0

001 = 1

etc.

1 = SET  
0 = RESET

### MSM5832 CLOCK CHIP

The following table describes the various registers in the MSM5832 and their limits:

REGISTER ADDRESS	COUNTER	DATA I/O				LIMITS	NOTES
		D3	D2	D1	D0		
0	S1	X	X	X	X	0 - 9	1
1	S10		X	X	X	0 - 5	1
2	M11	X	X	X	X	0 - 9	
3	M110		X	X	X	0 - 5	
4	H1	X	X	X	X	0 - 9	
5	H10	+	+	X	X	0 - 2	2
6	W		X	X	X	0 - 6	3
7	D1	X	X	X	X	0 - 9	
8	D10		+	X	X	0 - 3	4
9	MO1	X	X	X	X	0 - 9	
A	MO10				X	0 - 1	
B	YR1	X	X	X	X	0 - 9	
C	YR10	X	X	X	X	0 - 9	

## NOTES

- (1) Register set to zero when written to.
- (2) D3=1 for 24hr or D3=0 for 12hr clock  
D2=1 for PM or D2=0 for AM
- (3) Days from Sunday. Sunday = 0, Monday =1, etc.
- (4) D2=1 for leap year.

## AN EXAMPLE

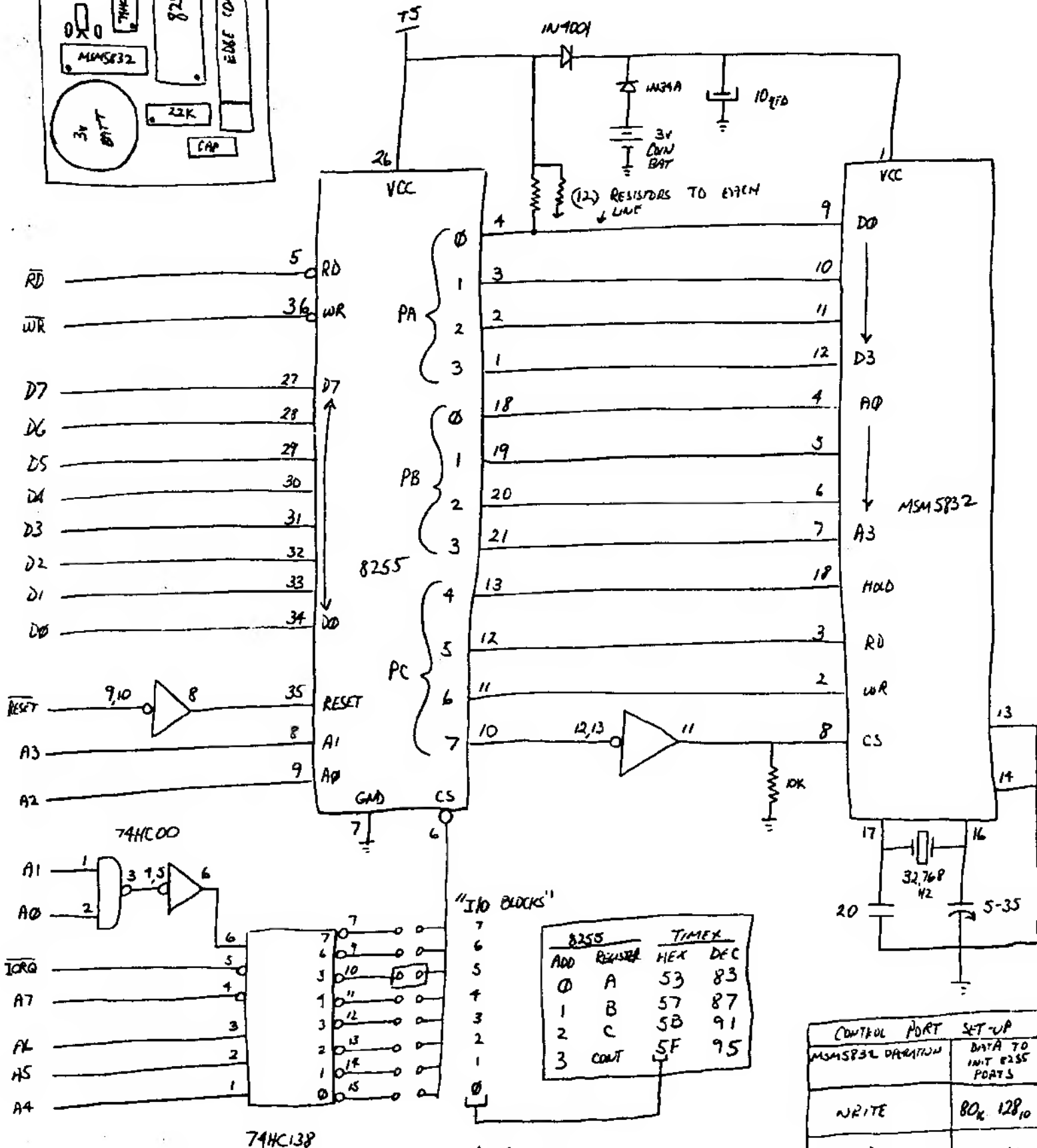
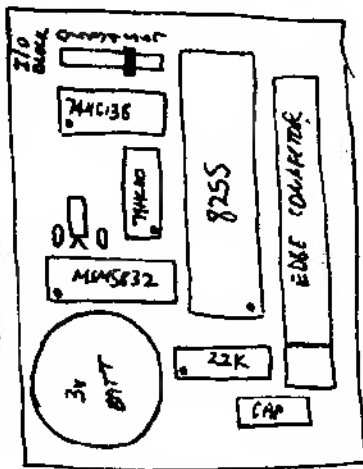
The following is an example showing how to read the day of the week from the clock module. Assume the module is mapped in block #5 ( ports 53, 57, 5B, 5F ).

- (1) After powering up the computer write a 90<sub>h</sub> to the 8255 CONTROL port, 5F<sub>h</sub>. This sets up the module to read the MSM5832 clock chip.
- (2) As a one-time operation clear port "C" in the 8255 by writing a 0 to port 5B<sub>h</sub>. This need only be done one time after powering up.
- (3) Write the desired MSM5832 register address to port "B" of the 8255 by writting a 6 ( 6 = day of the week ) to port 57<sub>h</sub>.
- (4) Strobe the MSM5832 RD signal active by writting a 0B<sub>h</sub> to the 8255's CONTROL port 5F<sub>h</sub>.
- (5) Read the day of the week by executing an input instruction on port 53<sub>h</sub>.
- (6) Return the MSM5832 RD signal inactive by writting a 0A<sub>h</sub> to port 5F<sub>h</sub>.
- (7) You can now read any other MSM5832 register by following steps 3, 4, 5, and 6 using the desired MSM5832 register address in step 3.

To change the day of the week to Saturday (6):

- (1) Initialize the 8255 to write the MSM5832 by writting an 80<sub>h</sub> to the 8255's CONTROL port 5F<sub>h</sub>.
- (2) Select the day of the week register in the MSM5832 by writting a 6 to port 57<sub>h</sub>.
- (3) Latch the new day of the week in the 8255's port A by writting a 6 ( Saturday ) to port 53<sub>h</sub>.
- (4) Store this new data in the MSM5832 by toggeling the WR signal. This is done by writting a 0D<sub>h</sub> to port 5F<sub>h</sub>. Then immediately follow by writting a 0C<sub>h</sub> to port 5F<sub>h</sub>.
- (5) You can now change any other MSM5832 register by writting the new register address to port 57<sub>h</sub>, latching the new data via port 53<sub>h</sub>, and toggling the MSM5832's WR signal as in step 4.

# ZX81/TSXXXX OUT-BOARD CLOCK MODULE



"I/O BLOCKS"

8255		TIMEEX	
ADD	REGISTER	HEX	DEC
0	A	53	83
1	B	57	87
2	C	5B	91
3	CONT	5F	95

CONTROL PORT	SET-UP
MSM5832 OPERATION	DATA TO INIT F255 PORTS
WRITE	80 <sub>h</sub> 128 <sub>10</sub>
READ	90 <sub>h</sub> 144 <sub>10</sub>

LIST 2068 TAPES of Public Domain Software

[illegible]

**TO ORDER:**  
Tapes - \$6.00 ea.  
Send a quality 60  
minute tape with a  
program on it and  
pay only \$3 each.  
**OR** attend meetings  
and pick up tapes  
for \$1.50 each!

Tape 9 and QL PLEASE let Harvey R.  
Tapes listed in next Know in advance which  
issue. [ ] Tape(s) you want.

Make checks payable to L.I.S.T.  
Send orders to:

Harvey Rair  
5 Peri Lane  
Valley Stream, NY 11581

Listings made with "Multitape"  
from tape #5A, program 16.

Whereas Topic 1, 2, or 3 and site 5 R21